Automated Review Rating System

Imbalanced model

Contents

ABSTRACT

The Automated Review Rating System (Imbalanced Model) focuses on predicting product or service ratings (1–5 stars) from customer review text using Natural Language Processing (NLP) and Machine Learning. In this version, the imbalanced dataset was created by using the remaining reviews after extracting the balanced dataset. The leftover data naturally contained unequal class distributions, reflecting real-world scenarios where certain ratings occur more frequently than others. The reviews were preprocessed, vectorized using TF-IDF, and classified with various machine learning models. This version highlights how class imbalance can lead to biased predictions, favoring majority classes while underperforming on minority ones. The imbalanced model serves as a baseline for comparison against the balanced version, illustrating the importance of dataset balance in improving accuracy and overall model fairness.

GitHub Setup

The complete project has been uploaded to GitHub repository for version control and easy access. This repository contains all the files related to the Automated Review Rating System. The repository also contains the documentation explaining each step of the project.

GitHub Repository Link: *Automated Review Rating System*

Environmental Setup

For implementing the Automated Review Rating System project, the environment was set up using Anaconda and Jupyter Notebook. Anaconda was used because it provides an easy way to manage Python libraries and virtual environments. The project was developed using python with the following main libraries installed:

- Python 3.12 (through Anaconda)
- Jupyter Notebook – for writing and running code interactively
- Pandas – for data handling and cleaning
- NumPy – for numerical operations
- Matplotlib and Seaborn – for data visualization
- Scikit-learn (sklearn) – for preprocessing, vectorization (TF-IDF), and train-test splitting
- NLTK – for basic text preprocessing

Data Collection

In this project, data was collected from online source Kaggle. The collected data included customer reviews and their corresponding ratings from different domains like Amazon products, hotels, and restaurants (Zomato). These datasets were chosen to get a wide variety of review texts and sentiments. All the datasets were combined into a single file containing three main columns as; reviews, overall rating, and review length.

Datasets Used:

1. **Amazon Review Dataset**

   Source: amazon-review

   The dataset contains product reviews posted by customers on Amazon. Each review includes a detailed description of the product experience and an overall rating from 1 to 5. Each record in the dataset includes detailed feedback from customers, aiding in product evaluation and market research.

2. **Book Reviews on Amazon**

Source: trending-books-reviews

This dataset offers an in-depth look into Amazon's top 100 Bestselling books along with their customer reviews, Ratings, Price etc.

3. **TripAdvisor Hotel Reviews**

Source: tripadvisor-hotel-reviews

This is a dataset of hotel reviews from the popular booking website TripAdvisor. This dataset contains 6,444 hotel reviews, each rated on a five-star scale.

4. **Zomato Reviews & Rating Dataset**

Source: zomato-reviews-ratings

This dataset contains restaurant reviews collected from Zomato. Each entry includes a textual review and a corresponding rating (1–5). Helpful in understanding text sentiment in the food service domain.

5. **Amazon Fine Food Reviews**

Source: amazon-fine-food-reviews

This dataset consists of reviews of fine foods from amazon. The data span a period of more than 10 years, including all ~500,000 reviews. Reviews include product and user information, ratings, and a plain text review. Additionally collected to balance the datasets.

These datasets are merged together and formed the final dataset to preprocess and visualize. The dataset contains raw, uncleaned reviews with missing values, duplicate entries, and mixed rating formats. Visualizations are performed on this uncleaned dataset to understand initial patterns and distributions before preprocessing.

The structure of dataset before cleaning:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30934 entries, 0 to 30933
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   review text    30929 non-null  object
 1   overall        30934 non-null  int64
 2   review_length  30934 non-null  int64
dtypes: int64(2), object(1)
memory usage: 725.1+ KB
```

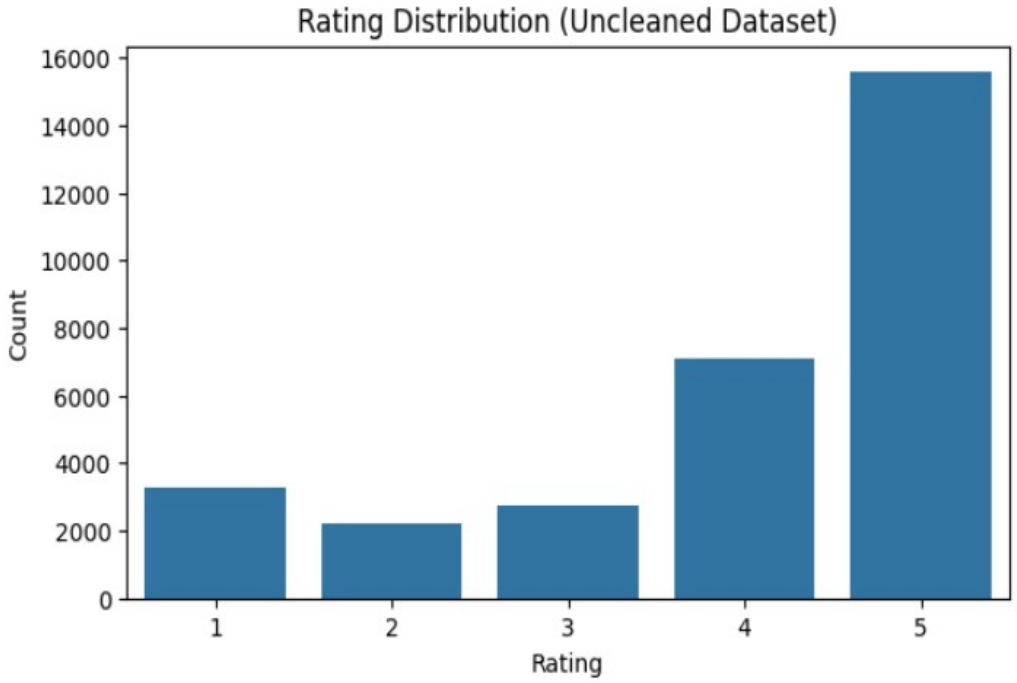
	review text	overall	review_length
0	No issues.	4	2
1	Purchased this for my device, it worked as adv...	5	31
2	it works as expected. I should have sprung for...	4	31
3	This think has worked out great.Had a diff. br...	5	66
4	Bought it with Retail Packaging, arrived legit...	5	52

Data Visualisation of Uncleaned dataset

Representing data graphically to uncover patterns, trends, and insights that are not immediately obvious in raw data.

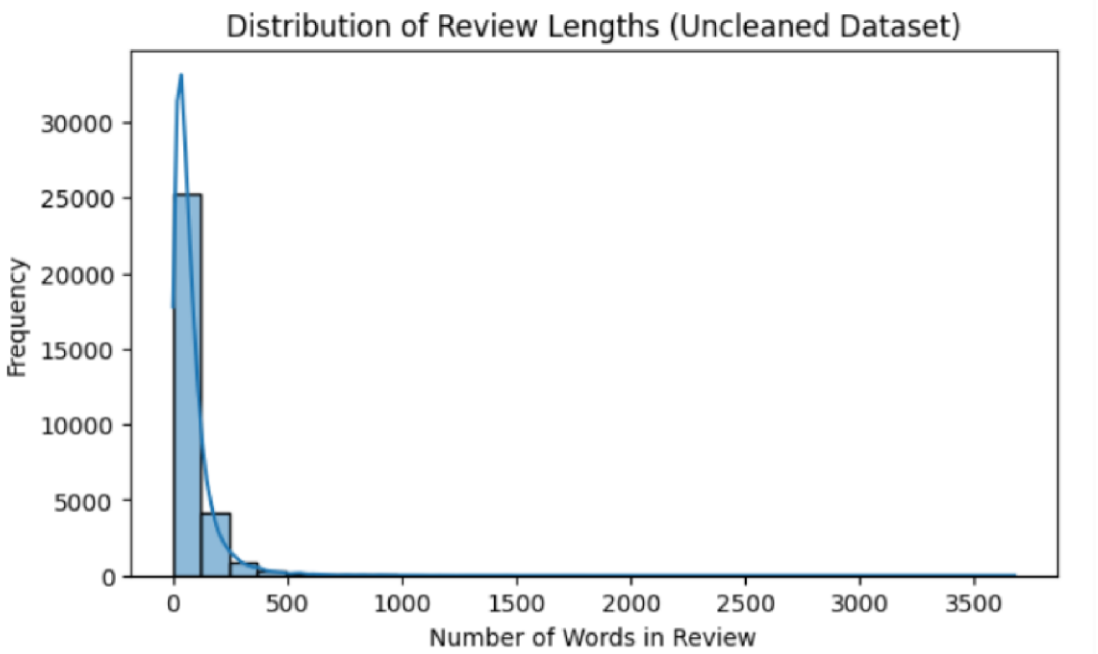
1) *Rating Distribution (barchart):*

```
plt.figure(figsize=(7,4))
sns.countplot(x='overall', data=df)
plt.title('Rating Distribution (Uncleaned Dataset)')
plt.xlabel('Rating')
plt.ylabel('Count')
plt.show()
```



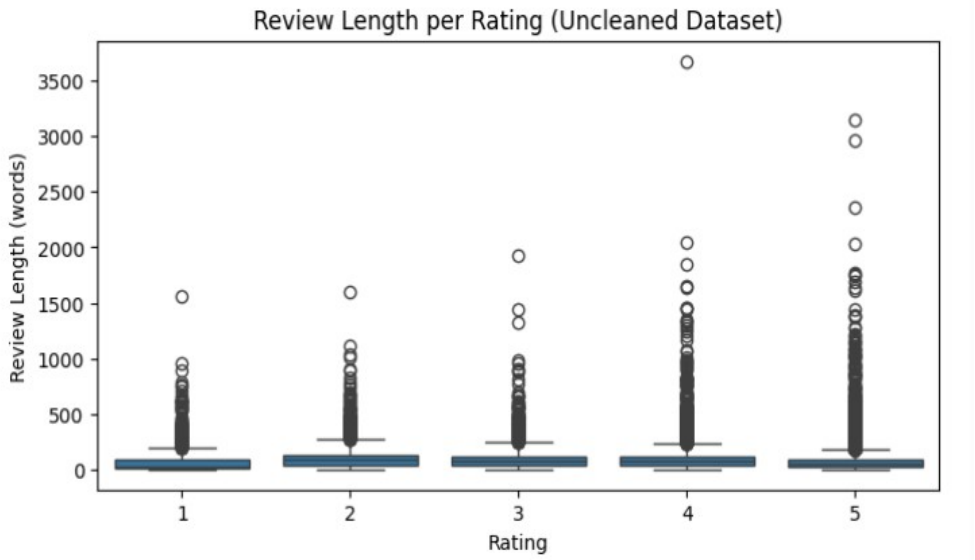
2) *Review length distribution (Histogram):*

```
plt.figure(figsize=(7,4))
sns.histplot(df['review_length'], bins=30, kde=True)
plt.title('Distribution of Review Lengths (Uncleaned Dataset)')
plt.xlabel('Number of Words in Review')
plt.ylabel('Frequency')
plt.show()
```



3) *Boxplot - Review Length per Rating :*

```
plt.figure(figsize=(8,4))
sns.boxplot(x='overall', y='review_length', data=df)
plt.title('Review Length per Rating (Uncleaned Dataset)')
plt.xlabel('Rating')
plt.ylabel('Review Length (words)')
plt.show()
```



Data Cleaning / Preprocessing

After merging all five datasets, the combined dataset underwent several preprocessing steps to ensure the data is clean, consistent, and ready for analysis.

- This involved removing null and duplicate entries, converting text to lowercase, and eliminating unwanted characters such as punctuation, numbers, special symbols, URLs, and extra spaces. After cleaning, the dataset became more consistent and suitable for visualization and further analysis.

1. Missing values:

To identify and remove any missing or incomplete records that could affect the analysis. There's 5 missing values in review column.

```
#missing values
print("\nMissing Values per Column:")
print(df.isnull().sum())
```

output:

```
Missing Values per Column:
review text   5
overall       0
dtype: int64
```

To remove:

```
df = df.dropna (subset=['review text', 'overall'])
```

Output

```
Missing values:
review text      0
overall          0
review_length    0
dtype: int64
```

2 Removing Duplicates:

Duplicate records can significantly affect the quality and accuracy of data analysis. After handling missing values, the dataset was analyzed for duplicate entries. Duplicates were categorized into two types:

- **Exact Duplicates:** Records with identical review text and rating. In this dataset there is no exact duplicates if any found they were removed immediately.
- **Conflicting Duplicates:** Records that share the same review text but have different ratings. There is no need to remove them.

```
exact_duplicates = df[df.duplicated(subset=['review text', 'overall'], keep=False)]
print(f" Exact duplicates found: {len(exact_duplicates)}")
conflicting_duplicates = (df[df.duplicated(subset=['review text'], keep=False)].sort_values('review text'))
print(f" Conflicting duplicates found: {len(conflicting_duplicates)}")
```

output: Exact duplicates found: 0
Conflicting duplicates found: 679

3. Text Cleaning:

To normalize and standardize the review text so that it becomes suitable for NLP-based analysis and model training.

This step removes noise, corrects formatting, and prepares a clean, meaningful representation of each review.

Steps involved:

- Lowercase all text
- Remove URLs, HTML tags, emojis, punctuation, and special characters
- Remove stopwords (using NLTK or SpaCy)
- Apply Lemmatization or Stemming (any one)
- Filter out reviews with fewer than 3 words and excessively long text

Necessary libraries needed:

import re ⟶ for Regular expressions (pattern matching, text cleaning)

```
def clean_text(text):
text = text.lower()
text = re.sub(r"http\S+|www\S+", "", text)
text = re.sub(r"<.*?>", "", text)
text = re.sub(r"[^a-z\s]", " ", text)
text = re.sub(r"\d+", " ", text)#numbers
text = text.encode('ascii', 'ignore').decode('utf-8')
text = re.sub(r"\s+", " ", text).strip() #extra spaces
return text
```

applying to review column:

```
df['reviews']=df['review text'].apply(clean_text)
```

Output:

	review text	overall	review_length	reviews
0	No issues.	4	2	no issues
1	Purchased this for my device, it worked as adv...	5	31	purchased this for my device it worked as adve...
2	it works as expected. I should have sprung for...	4	31	it works as expected i should have sprung for ...
3	This think has worked out great.Had a diff. br...	5	66	this think has worked out great had a diff bra...
4	Bought it with Retail Packaging, arrived legit...	5	52	bought it with retail packaging arrived legit ...

After this step, the existing review column is dropped and the review lengths are recalculated based on the cleaned reviews after preprocessing.

```
df = df.drop (columns = ['review text'])
df ['review_length'] = df ['reviews'].str.split().apply(len)
```

Output:

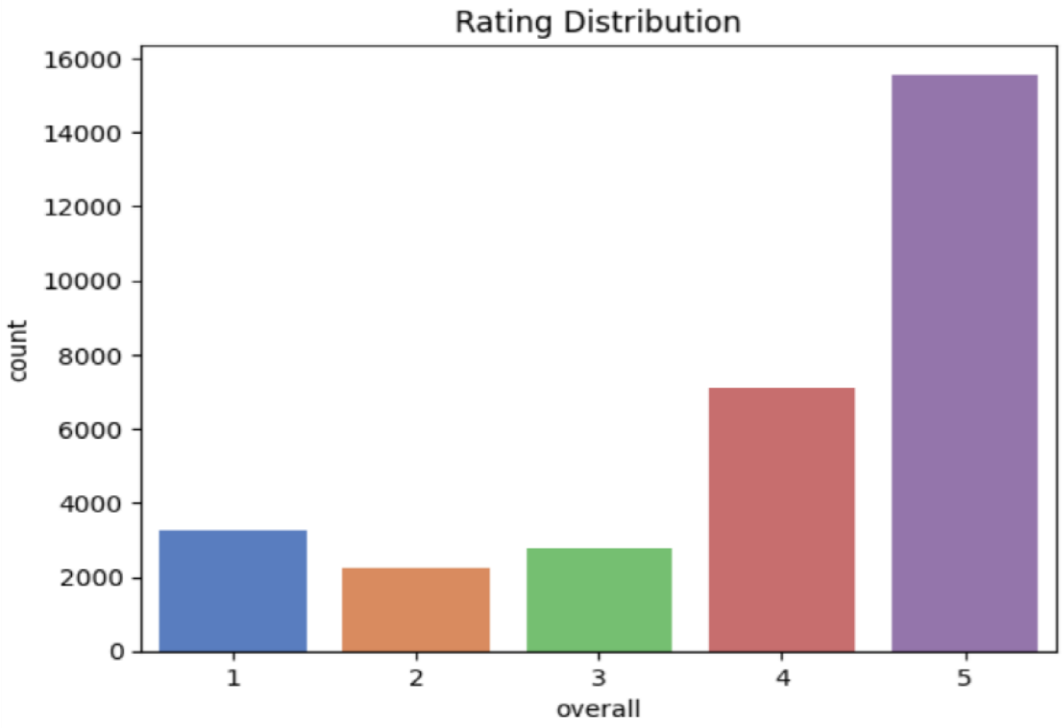
	overall	review_length	reviews
0	4	2	no issues
1	5	31	purchased this for my device it worked as adve...
2	4	31	it works as expected i should have sprung for ...
3	5	70	this think has worked out great had a diff bra...
4	5	51	bought it with retail packaging arrived legit ...

Visualization of cleaned dataset

To analyze how the dataset looks after cleaning and check if the data distribution remains consistent using seaborn.

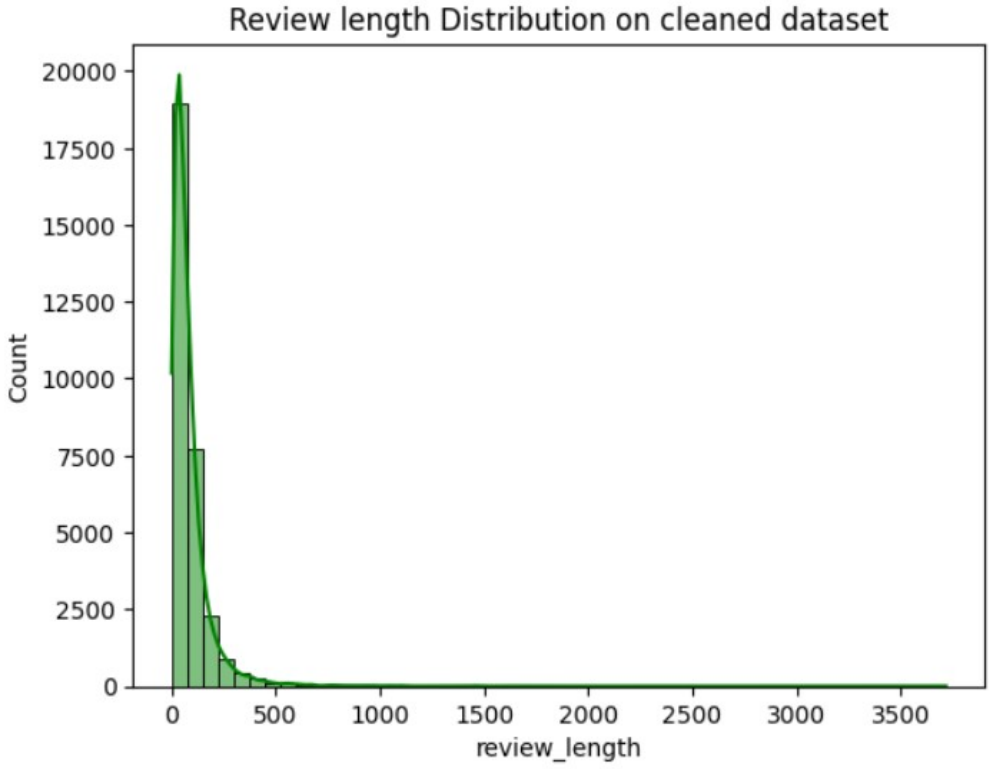
1) *Rating Distribution (Bar Chart)*

```
sns.countplot(x="overall", data=df, hue="overall", palette="muted",
legend=False)
plt.title("Rating Distribution")
plt.show()
```



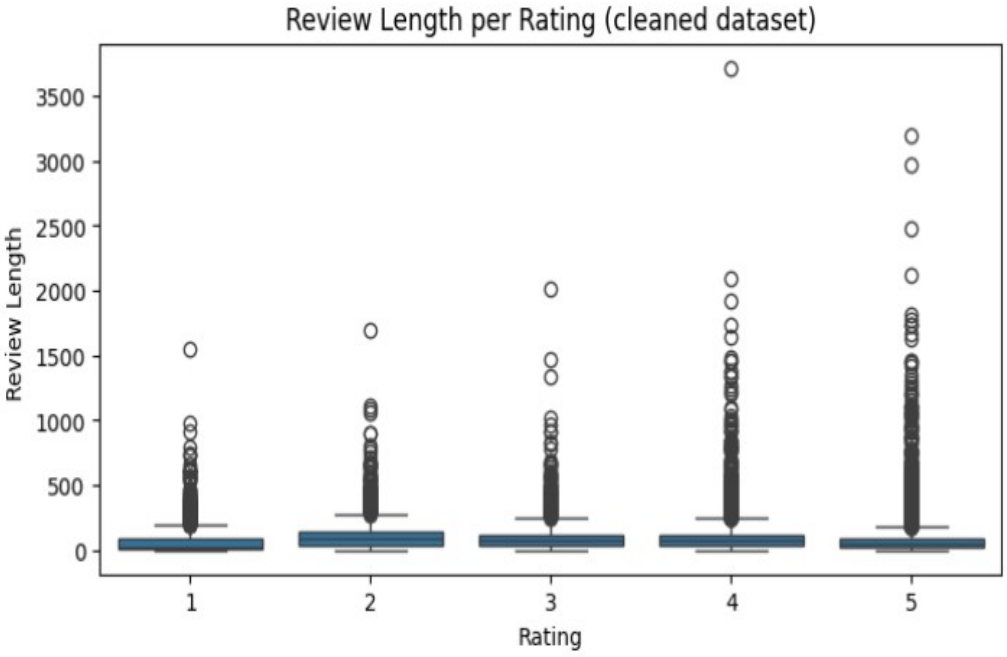
2) *Review length distribution (histogram)*

```
#histogram using seaborn
sns.histplot(df['review_length'], bins=50, kde=True, color="green")
plt.title("Review length Distribution on cleaned dataset")
plt.show()
```



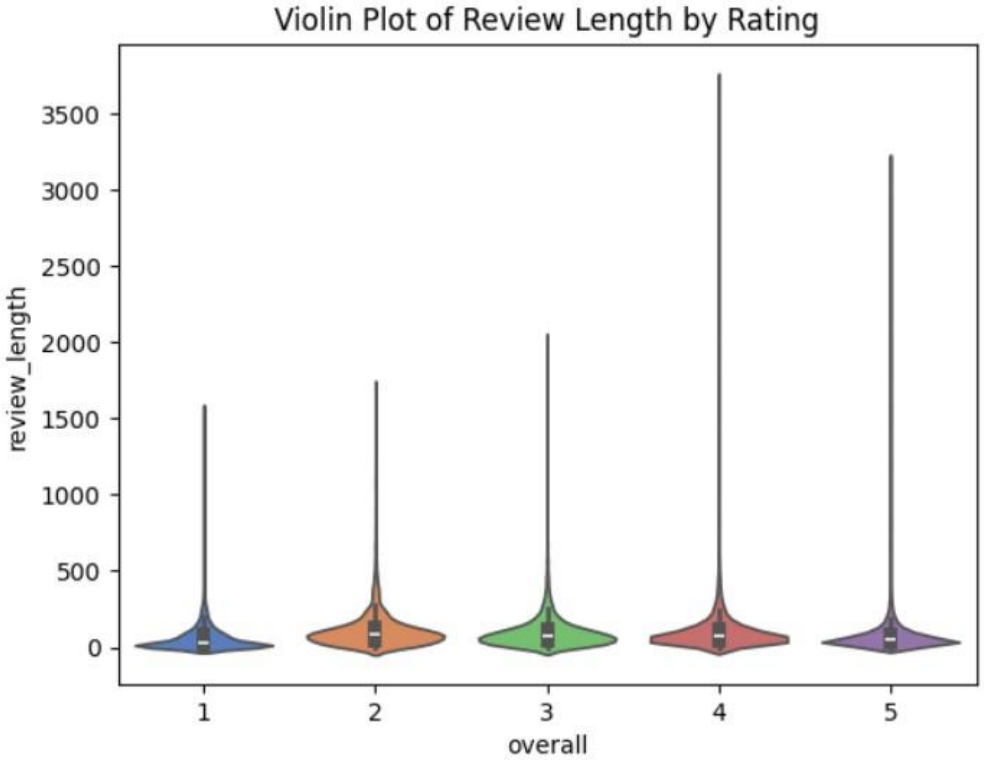
3) *Boxplot of Review Length per Rating:*

```
plt.figure(figsize=(8,4))
sns.boxplot(x='overall', y='review_length', data=df)
plt.title('Review Length per Rating (cleaned dataset)')
plt.xlabel('Rating')
plt.ylabel('Review Length ')
plt.show()
```



4) *Violin Plot (Density + Distribution)*

```
sns.violinplot(x='overall', y='review_length', data=df, hue='overall',
palette="muted", legend=False)
plt.title("Violin Plot of Review Length by Rating")
plt.show()
```



IMBALANCED DATA CREATION

After creating the balanced dataset, the remaining samples (those not included in the balanced set) were taken as the imbalanced dataset. This reflects the natural distribution of ratings in the original data. The leftover data was extracted using the drop() function and saved as a separate DataFrame named imbalanced_df. And later it is splitted by ratio as 10% , 15%, 20%, 25%, 30% for each rating.

Code:

```
leftover_df = df.drop(balanced_df.index).reset_index(drop=True)
imbalanced_df = leftover_df.copy()
```

Structure of Imbalanced data:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20929 entries, 0 to 20928
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   review text    20929 non-null  object
 1   overall        20929 non-null  int64
 2   review_length  20929 non-null  int64
dtypes: int64(2), object(1)
memory usage: 490.7+ KB
```

	review text	overall	review_length
0	The Delivery from zomato was amazing!!!Deliver...	5	13
1	Delivery guy was rude	4	4
2	Cheaters. They dont deliver what you see in th...	1	32
3	poor. the pic showed broccoli in chicken meal....	4	22
4	ur sandwiches never disappoint me..they taste ...	2	25

	review text	overall	review_length
20924	best kept secret 3rd time staying charm \tnot ...	5	109
20925	great location price view hotel great quick pl...	4	39
20926	ok just looks nice modern outside \tdesk staff...	2	63
20927	hotel theft ruined vacation hotel opened sept ...	1	781
20928	people talking \tca n't believe excellent rati...	2	90

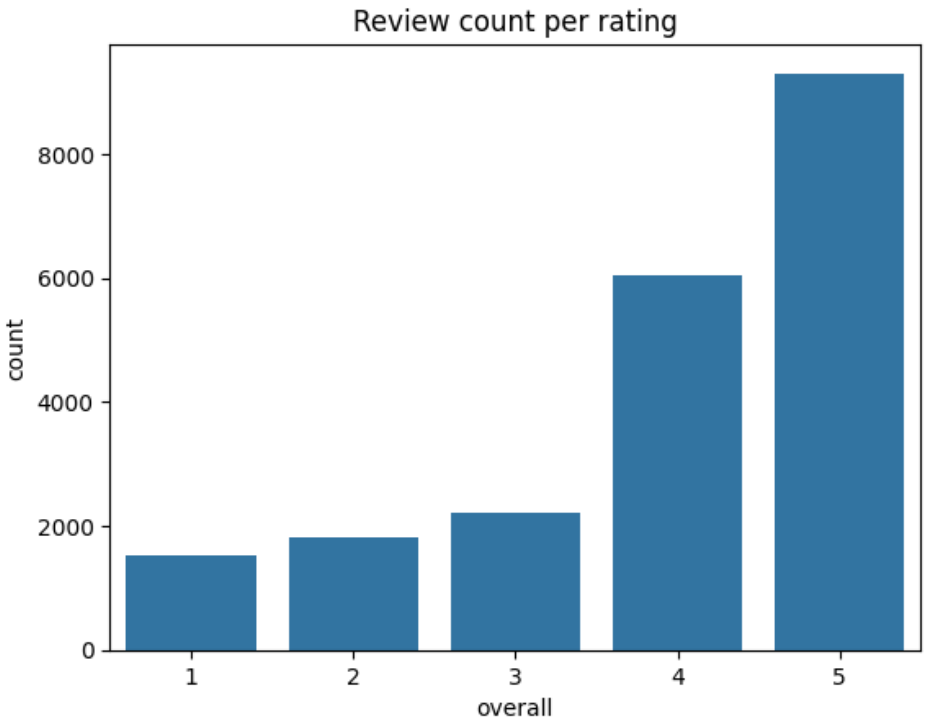
IMBALANCED DATA VISUALISATION

To understand the distribution of review ratings and identify imbalance across classes, visualization techniques were applied. Using libraries like Matplotlib and Seaborn, bar charts were created to display the count of reviews for each rating (1–5).

1) *Bar Plot Visualization*

To visualize the distribution of review ratings in the imbalanced dataset

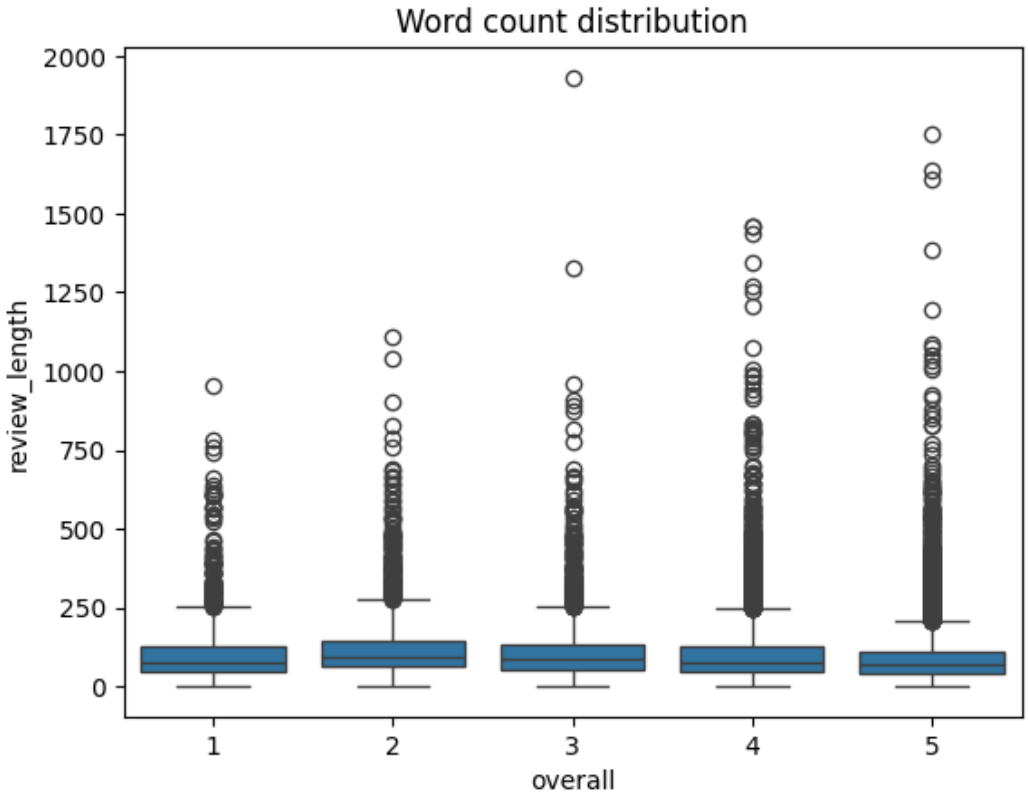
```
sns.countplot(x="overall", data=imbalanced_df)
plt.title("Review count per rating")
plt.show()
```



2) *Box Plot Visualization*

To visualize the variation in review lengths across different rating categories.

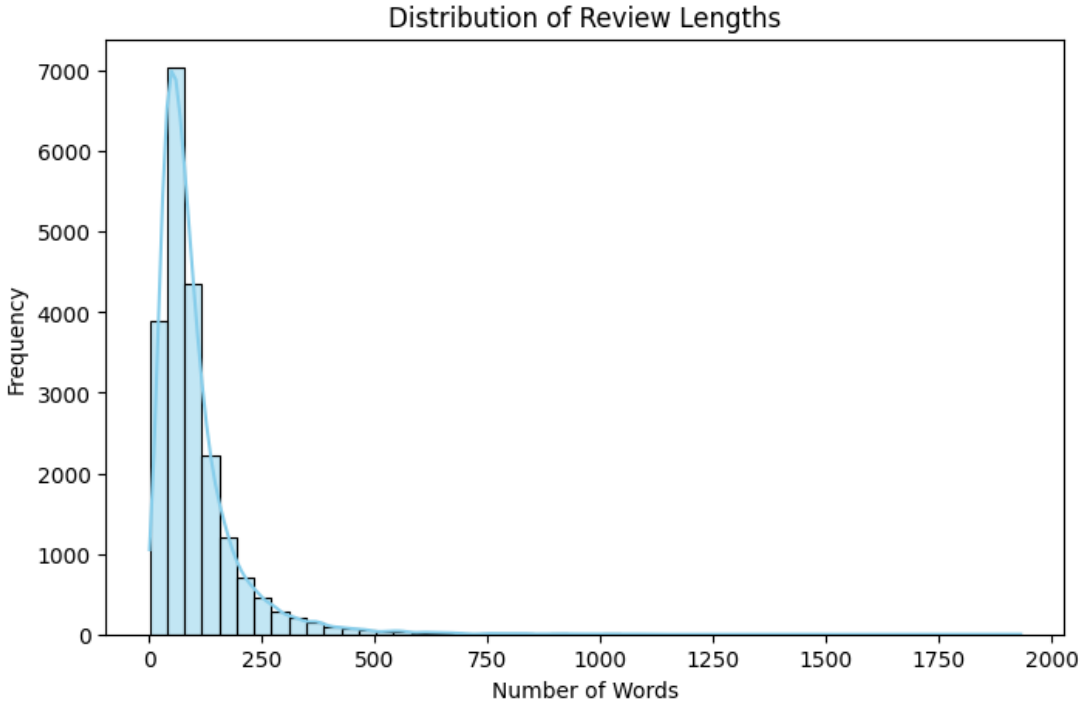
```
sns.boxplot(x="overall", y="review_length", data=imbalanced_df)
plt.title("Word count distribution")
plt.show()
```



3) *Review Length Distribution*

To analyze how review lengths are distributed in the imbalanced dataset.

```
plt.figure(figsize=(8,5))
sns.histplot(imbalanced_df['review_length'], bins=50, kde=True, color='skyblue')
plt.title('Distribution of Review Lengths')
plt.xlabel('Number of Words')
plt.ylabel('Frequency')
plt.show()
```



Tokenization and Stopword Removal:

Stopwords are common words (e.g., *"the"*, *"is"*, *"and"*) that do not carry significant meaning and for this convert each review into individual words (tokens) and remove common, noninformative words(stopwords) to focus on meaningful text for analysis and NLP tasks.

NLTK (Natural Language Toolkit**)** is used for tokenization and stopword removal.

For tokenizing:

```
#spliting string to words
imbalanced_df['words']=imbalanced_df['reviews'].apply(lambda x: x.split())
```

This convert the string of words in review column to token of words.

Eg: no issues ⟶ [no, issues]

Output:

	overall	review_length	reviews	words
0	5	13	the delivery from zomato was amazing delivery ...	[the, delivery, from, zomato, was, amazing, de...
1	4	4	delivery guy was rude	[delivery, guy, was, rude]
2	1	32	cheaters they dont deliver what you see in the...	[cheaters, they, dont, deliver, what, you, see...
3	4	22	poor the pic showed broccoli in chicken meal i...	[poor, the, pic, showed, broccoli, in, chicken...
4	2	25	ur sandwiches never disappoint me they taste j...	[ur, sandwiches, never, disappoint, me, they, ...

Necessary libraires:

```
Import nltk
from nltk.corpus
import stopwords
```

These libraries from NLTK (Natural Language Toolkit**)** are used for text preprocessing in natural language processing tasks and downloaded stopwords.

```
nltk.download('stopwords')
```

Loading English stopwords and printing them:

```
stop_words = set(stopwords.words('english'))
print("Total Stopwords:", len(stop_words))
print("Stopwords List:", stop_words)
```

output:

```
Total Stopwords: 198
Stopwords List: {'s', "we're", 'how', 'myself', 'yourself', 'themselves', 'this', 'ours', 'shan
', 'these', 'the', 'because', 'other', 'who', 'further', 'where', 'than', 'have', 're', 'agains
t', 'own', 'during', 'was', 'you', "she'd", 'aren', 'himself', "i'll", "doesn't", 'each', 'hasn
', "should've", 'what', 'through', 'no', 'below', 'will', 'with', 'did', 'some', 'all', 'itself
', 'their', 'didn', "he'll", 'then', "they'll", "they're", 'from', "don't", 'its', "you'll", "t
hat'll", 've', 'your', 'only', 'wouldn', 'i', 'as', 'nor', 'by', 'were', 'haven', 'had', 'it',
"won't", 'y', 'so', 'doing', 'o', "we've", "it'll", 'again', "i've", "you've", 'm', "he'd", 'mu
stn', 'whom', 'mightn', 'me', "mustn't", "isn't", 'between', "shan't", "wouldn't", 'about', "co
uldn't", 'can', 'll', 'couldn', 'most', 'been', "it's", 'under', "we'd", 'isn', 'too', 'ain', '
both', "i'd", 'needn', "you'd", 'any', 'an', 'our', 'there', 'he', 'don', "hadn't", 'am', 'just
', 'my', 'at', 'that', 'not', 'to', 'on', 'hers', 'when', "she'll", "wasn't", 'being', 'do', "h
e's", 'wasn', 'we', 'having', "hasn't", 'and', "didn't", 'weren', 'up', "mightn't", 'herself',
'above', "weren't", 'which', 'before', 'shouldn', "we'll", 'while', 'for', 'be', "shouldn't", '
those', 'of', 'why', "it'd", 'now', 'after', 'ourselves', "she's", 'won', 'doesn', "haven't", '
him', 'more', 'his', "aren't", 'or', 'd', 'ma', 'such', 'yours', 'off', 'same', 'theirs', 'them
', 't', 'into', 'should', 'very', 'if', 'down', 'but', 'until', "they've", 'she', 'has', 'over'
, 'here', 'is', 'they', 'a', 'hadn', 'once', 'does', 'in', 'are', "you're", 'yourselves', 'few'
, 'her', "needn't", 'out', "i'm", "they'd"}
```

Stopwords removal:

```
def clean_words_list(words):
 words = [w for w in words if w not in stop_words]
 return " ".join(words)
```

After removing stopwords from the words column, the cleaned tokens were rejoined into strings and saved back into the reviews column for further text processing.

```
imbalanced_df['reviews'] = imbalanced_df['words'].apply(clean_words_list)
```

output:

	overall	review_length	reviews	words
0	5	13	delivery zomato amazing delivery man also help...	[the, delivery, from, zomato, was, amazing, de...
1	4	4	delivery guy rude	[delivery, guy, was, rude]
2	1	32	cheaters dont deliver see pic ordered chicken ...	[cheaters, they, dont, deliver, what, you, see...
3	4	22	poor pic showed broccoli chicken meal wasnt in...	[poor, the, pic, showed, broccoli, in, chicken...
4	2	25	ur sandwiches never disappoint taste superb pl...	[ur, sandwiches, never, disappoint, me, they, ...

Why NLTK Was Used?

- NLTK is lightweight and ideal for basic preprocessing tasks like stopword removal, tokenization, and lemmatization.
- SpaCy, while more powerful and faster for large-scale NLP tasks, requires additional model downloads and setup.

- NLTK was preferred for its simplicity, efficiency and ease of integration.

Lemmatization:

Lemmatization is a text preprocessing technique used in Natural Language Processing (NLP) to reduce words to their base or root form, known as a lemma.

example: words like *"running"*, *"ran"*, and *"runs"* are all converted to their base form *"run"*

Necessary libraries:

```
import pandas as pd
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet
```

NLTK downloads:

```
nltk.download('wordnet')
nltk.download('omw-1.4')
nltk.download('punkt')
nltk.download('punkt_tab')
```

Initialising lemmatized and applying lemmatization on tokens:

```
lemmatizer = WordNetLemmatizer()
def clean_words_list(text):
   from nltk.tokenize import word_tokenize
   words = word_tokenize(str(text))
   lemmatized = [lemmatizer.lemmatize(w) for w in words]
   return " ".join(lemmatized)
```

Apply back:

```
imbalanced_df['lemmatized_words'] =imbalanced_df['reviews'].apply(clean_words_list)
```

output:

	overall	review_length	reviews	words	lemmatized_words
0	5	13	delivery zomato amazing delivery man also help...	[the, delivery, from, zomato, was, amazing, de...	delivery zomato amazing delivery man also help...
1	4	4	delivery guy rude	[delivery, guy, was, rude]	delivery guy rude
2	1	32	cheaters dont deliver see pic ordered chicken ...	[cheaters, they, dont, deliver, what, you, see...	cheater dont deliver see pic ordered chicken m...
3	4	22	poor pic showed broccoli chicken meal wasnt in...	[poor, the, pic, showed, broccoli, in, chicken...	poor pic showed broccoli chicken meal wasnt in...
4	2	25	ur sandwiches never disappoint taste superb pl...	[ur, sandwiches, never, disappoint, me, they, ...	ur sandwich never disappoint taste superb pls ...

Check length of reviews vs lemmatized text

```
imbalanced_df['lemmatized_length'] = imbalanced_df['lemmatized_words'].apply(lambda x: len(str(x).split()))
imbalanced_df[ ['review_length','lemmatized_length']].head(10)
```

output:

	review_length	lemmatized_length
0	13	9
1	4	3
2	32	18
3	22	16
4	25	18
5	11	6
6	31	16
7	9	9
8	6	2
9	2	2

Dropping unwanted columns:

```
imbalanced_df = imbalanced_df.drop(columns=['words'])
imbalanced_df = imbalanced_df.drop(columns=['reviews'])
```

output:

	overall	review_length	lemmatized_words	lemmatized_length
0	5	13	delivery zomato amazing delivery man also help...	9
1	4	4	delivery guy rude	3
2	1	32	cheater dont deliver see pic ordered chicken m...	18
3	4	22	poor pic showed broccoli chicken meal wasnt in...	16
4	2	25	ur sandwich never disappoint taste superb pls ...	18

Count of each rating

```
print(imbalanced_df['overall'].value_counts())
```

```
overall
5    9305
4    6058
3    2226
2    1817
1    1523
Name: count, dtype: int64
```

Analysis of Short and Long Reviews (filtering)

The below code Filters reviews based on their lemmatized word count to separate very short and long reviews.

```
# Short reviews (length 3-50)

short_reviews = imbalanced_df[(imbalanced_df['lemmatized_length'] >= 0) &
(imbalanced_df['lemmatized_length'] <= 3)]
print("Short reviews:")
print(short_reviews['lemmatized_words'].head(10))

# Long reviews (length 151-200)

long_reviews = imbalanced_df[(imbalanced_df['lemmatized_length'] > 150) &
(imbalanced_df['lemmatized_length'] <= 200)]
print("\nLong reviews:")
print(long_reviews['lemmatized_words'].head(10))
```

Output

```
Short reviews:
1     delivery guy rude
8           spicy pizza
9       disgusting food
14      beautiful tesat
16           like taste
19      like taste idli
21           late order
26    fine may accident
33                worst
34                yummy
Name: lemmatized_words, dtype: object

Long reviews:
442    great stay great stay went seahawk game awesom...
458    service service service spent week g friend la...
466    nice hotel husband stayed warwick year ago lik...
483    great location n stay long needed place stay s...
540    absolutely charming good value stayed numerous...
548    fun charming needed issue stayed marqueen day ...
574    perfect love old hotel searching web place min...
576    noisy good value felt like weekend away wife k...
591    miss stayed hotel friend weekend th st septemb...
614    dirty place watch limited english used reading...
Name: lemmatized_words, dtype: object
```

Removing them:

```
imbalanced_df=imbalanced_df[(imbalanced_df['lemmatized_length']>=3)&(imbalanced_df['lemmatized_length']<=200)]
```

output:

	overall	review_length	lemmatized_words	lemmatized_length
0	5	13	delivery zomato amazing delivery man also help...	9
1	4	4	delivery guy rude	3
2	1	32	cheater dont deliver see pic ordered chicken m...	18
3	4	22	poor pic showed broccoli chicken meal wasnt in...	16
4	2	25	ur sandwich never disappoint taste superb pls ...	18

Current rating count

```
overall
5    8631
4    5455
3    1987
2    1571
1    1356
Name: count, dtype: int64
```

As there is no need for existing review length column therefore, it is dropped and renamed the columns names for easy convenience as follows;

```
imbalanced_df = imbalanced_df.drop(columns=['review_length'])
imbalanced_df.rename(columns={'lemmatized_words': 'reviews'}, inplace=True)
imbalanced_df.rename(columns={'lemmatized_length': 'review_length'}, inplace=True)
```

output

	overall	reviews	review_length
0	5	delivery zomato amazing delivery man also help...	9
1	4	delivery guy rude	3
2	1	cheater dont deliver see pic ordered chicken m...	18
3	4	poor pic showed broccoli chicken meal wasnt in...	16
4	2	ur sandwich never disappoint taste superb pls ...	18

Custom Imbalanced Dataset Creation

To simulate a more realistic rating distribution, a custom imbalanced dataset was created by sampling different proportions of reviews from each rating class. Instead of using an equal number of samples per class, varying amounts were chosen like, 10% from rating 1, 15% from rating 2, 20% from rating 3, 25% from rating 4, and 30% from rating 5. This approach reflects how users tend to give higher ratings more frequently in real world scenarios.

For this following code used:

```
samples = {
   1: 1000,   # 10%
   2: 1500,   # 15%
   3: 1987,   # 20%
   4: 2500,   # 25%
   5: 3000    # 30%
}
sampled_dfs = []

for rating, n in samples.items():
   subset = imbalanced_df[imbalanced_df['overall'] == rating]
   sampled = subset.sample(n=n, random_state=42)
   sampled_dfs.append(sampled)

df_imbalanced = pd.concat(sampled_dfs).sample(frac=1, random_state=42).reset_index(drop=True)
```

output:

```
overall
5    3000
4    2500
3    1987
2    1500
1    1000
Name: count, dtype: int64
Total rows: 9987
```

	overall	reviews	review_length
0	2	star lobby star hotel stayed start end europe ...	65
1	3	ok night understand people shout absolutely lo...	77
2	5	luxury valley wing stayed valley wing bedroome...	53
3	4	decent hotel great location friendly staff hot...	16
4	1	book hotel month arrived decribed building sit...	139
...	...	...	...
9982	4	great hotel location worried booking hotel rea...	58
9983	4	great place relax got night stay march check f...	110
9984	4	miramar excellent location miramar great locat...	62
9985	1	problem way stayed night property bad stayed r...	33
9986	5	toronto hidden gem tired old hotel room travel...	181

9987 rows × 3 columns

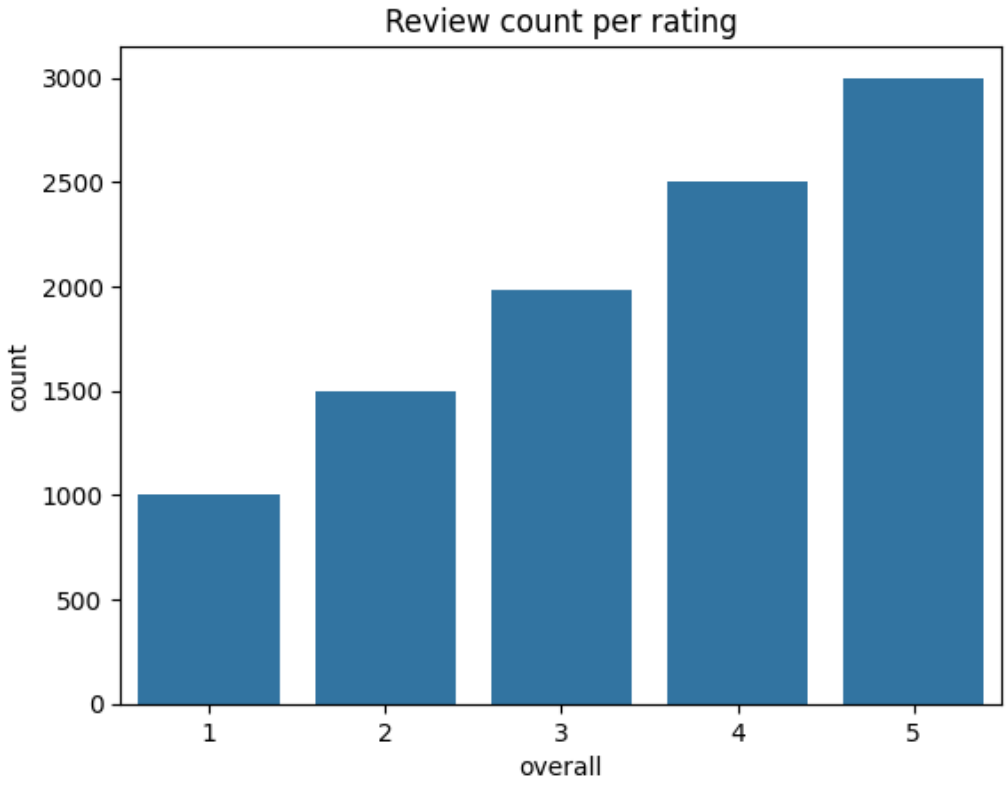
From rating 3 (which had slightly fewer available samples as it was an imbalanced set it was not considered.

Visualisation after Cleaning and splitting

After cleaning the text data and splitting it into training and testing sets, additional visualizations were performed to understand the distribution of reviews and check data consistency.

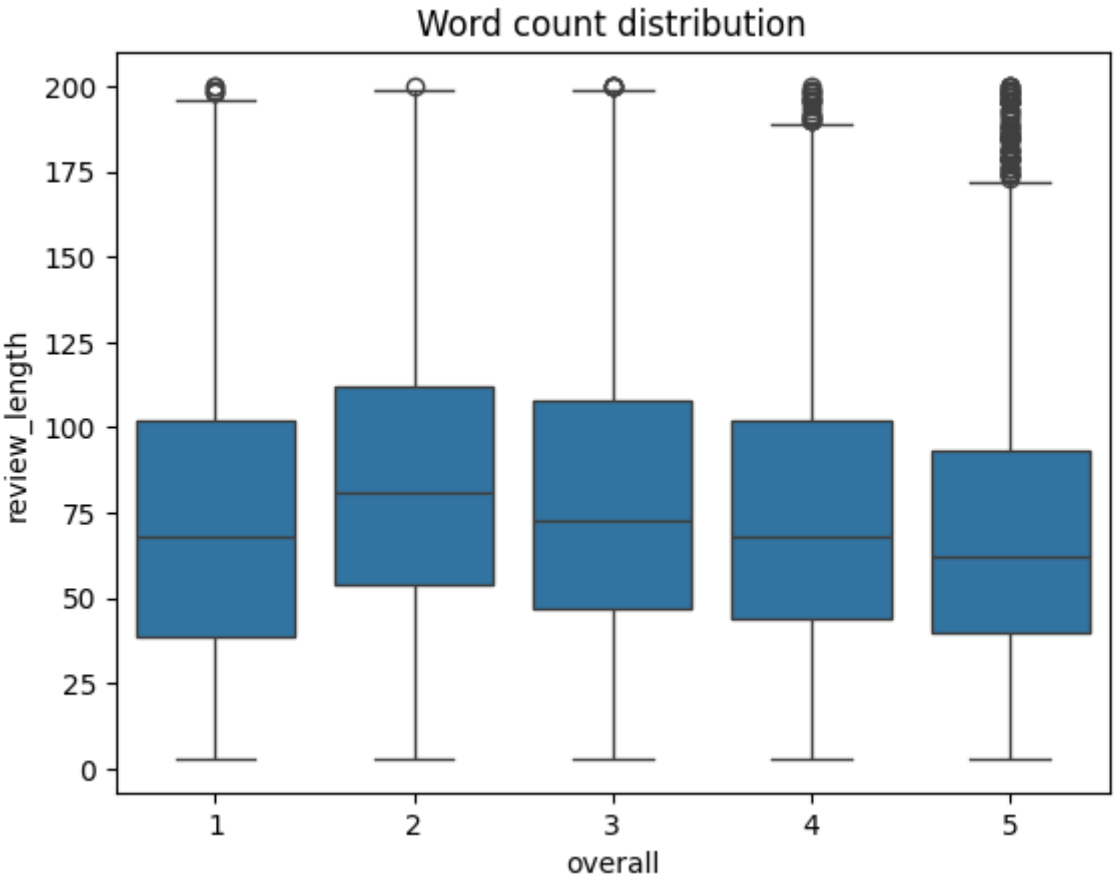
1) *Review Count per Rating*

```
sns.countplot(x="overall", data=df_imbalanced)
plt.title("Review count per rating")
plt.show()
```



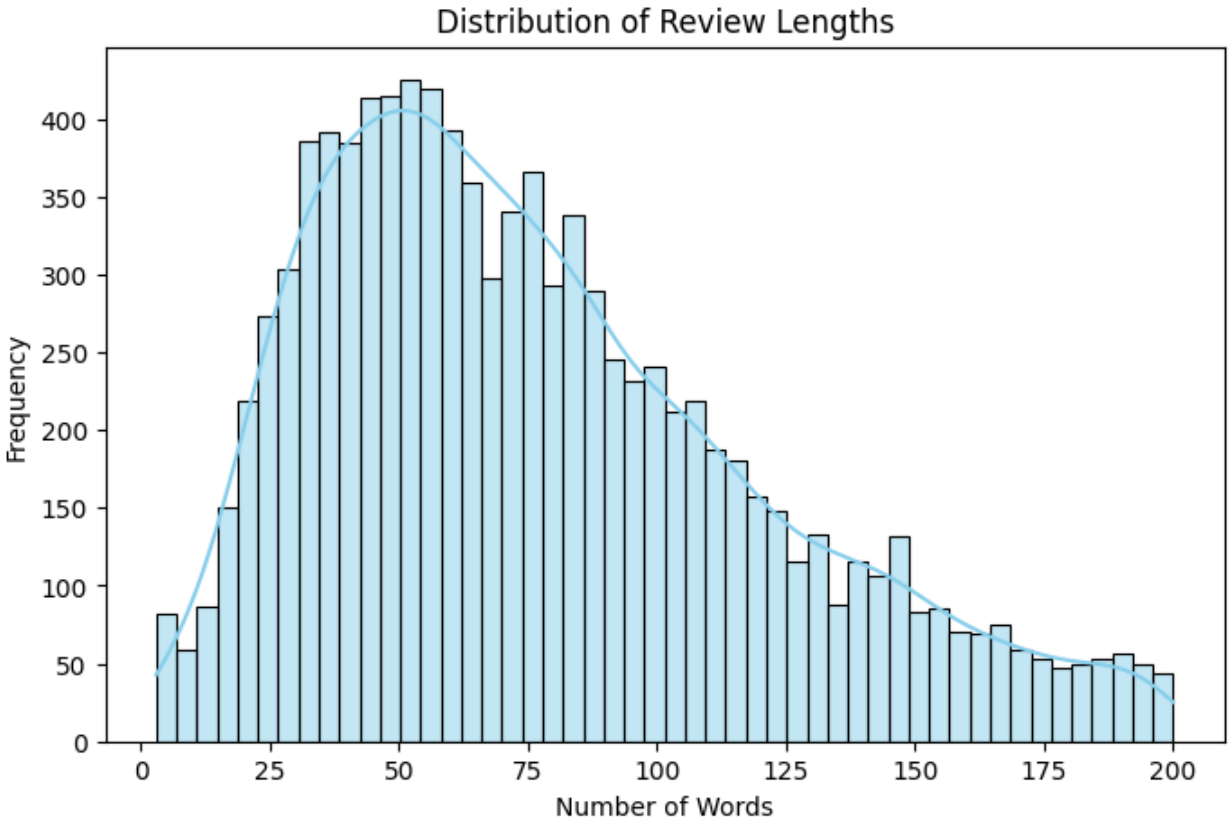
2) *Word Count Distribution*

```
sns.boxplot(x="overall", y="review_length", data=df_imbalanced)
plt.title("Word count distribution")
plt.show()
```



3) *Review Length Distribution*

```
plt.figure(figsize=(8,5))
sns.histplot(df_imbalanced['review_length'], bins=50, kde=True, color='skyblue')
plt.title('Distribution of Review Lengths')
plt.xlabel('Number of Words')
plt.ylabel('Frequency')
plt.show()
```



4) *displaying samples:*

```
for rating in sorted(df_imbalanced["overall"].unique()):
    print(f"\n⭐ Rating {rating} Sample Reviews:")
    print(df_imbalanced[df_imbalanced["overall"] == rating]["reviews"].head(4).to_list())
```

output

⭐Rating 1 Sample Reviews:

['book hotel month arrived decribed building site picked jaw ground asked builder going directed reception hard hat s upplied place dangerously derelict downstairs hotel manager seat bag cement manager explained refurbisments goin g reminded said thing phone previous day order went tell apartment available street away voiced utter dissappointm ent travelling seven month old baby adult tired took key apartment described follows flight stair elevator baby buggy fit tiny lift person walk flight buggy padlocked kitchen door electric faulty kitchen bound way feed baby management quite aggressive instance went complain given breakfast voucher little cafe beneath hotel stand breakfast sit table in curr charge managed muddle night purchased kettle make bottle reason booked hotel roof terrece invisenged wonde rful evening sipping wine taking breath taking view beautiful city offer instead stuck ramshackle apartment described mediocre good day receieve night free way apology highly dissatisfactory', 'good thing hotel location good thing hote l location paid extra city view room room overly clean staff unhelpful long wait uninterested concierge stay dissapoint ed', 'healthy soup panner meal perfect combo grt taste must eat', 'staff respect sleeping guest staff friendly helpful ro om small n really bother sleeping needed bed toilet got room basement right staff exit delivery entrance linen cupboa rd staff toilet night woken staff shouting running stair use toilet morning delivery door slammed shut shaking room b reakfast room directly room hear chair scraped floor finnally night alarm went second woke difficult sleep hear recept ionist talking alarm going n hotel fault normally lasted second set staff suspect room hotel basement hotel fine wo n staying risk worth']

⭐Rating 2 Sample Reviews:

['star lobby star hotel stayed start end europe vacation positive great location walking museum restaurant lobby bar breakfast expect good hotel room service wouldexpect budget hotel minimal lighting worn bedspread sheet loose sho wer head return c n working room staff fix saw room andstayed weekend row n mind staying budget hotel save buck

memphis priced budget hotel internet deal allows pay star price stay hotel', 'boyfriend stayed weekend booking layne unseen hotwire turned alright great terrible reviewer experience good decor place cute old fashioned cute bordered c reepy say man downstairs counter n friendly arrived quickly warmed quite nice ended checking little late n say charge extra room decent size clean microwave fridge bed nice springy walking distance union square really nice n say union square hotwire enjoy using old fashioned elevator n floor take long come really worth waiting bad neighborhood defi nitely sketchy room facing street lot noise late night pretty early morning stair pretty narrow curved n try wearing he el towel small wondered maybe forgotten bath towel one slightly larger hand towel say thing bothered place way be d sheet pulled way inch mattress liner mattress exposed foot bed basically remake bed beige colored blanket standar d place bunch hole n look moth bitten thankfully passable place stay tight budget great terrible', 'good long stair imp ossibly steep room tiny person stay bed let room recommend hotel', 'great location average service picked hotel locat ion trusted novotel reputation location great service standard expected novotel good subway station right outside ho tel get subway line major shopping area xidan silk market wanfujing major attraction forbidden city tienanmen squar e line changing line needed stayed standard room hotel n nice new clean roomy comfortable really hard bed bottle w ater provided free day average breakfast price hotel expensive china try deal included room package really clean goo d quality bread shop road ate breakfast time bad service average check staff slow paperwork changed hand whilst pu shed girl ticket desk extremely rude unhelpful talk phone answer question difficult help domestic air ticket booked flig ht china travel hotel people concierge desk uninterested unhelpful meant getting direction major attraction difficult p lan trip know want use confirm destination write address chinese taxi driver overall stay expect great service']

⋆Rating 3 Sample Reviews:
['ok night understand people shout absolutely loved american overstatement mario ok nothing cheer following experi ence night mario september pro small neat clean hotel staff ok breakfast better rich perfectly located florence confer ence visitor con smallest room seen italy bathroom model aircraft toilet shower room poor lightning ifficult read bed ask staff help safe deposit box bring torch air conditioning worked night daily complaint sewage smell bathroom plac e florence ok night euro night grateful wife share microscopic room', 'world best hyatt regency enjoy regent singapor e best temper expectation n think second season managed hotel town season singapore block away connected orcha rd road air conditioned passaged linked singapore hilton best hyatt regency world doubt note john portman designed hotel easily downtown atlanta bubble glass elevator corkscrew circular staircase ivy trellised sun dappled atrium cent er concrete building ala material bit better average garden variety hyatt atrium glowing brass fine dark wood gleami ng marble room suite quite bright floor ceiling window sliding glass door suite yellow wall pink exterior door bit odd t aste hotel actually great shape look renovated recently upkeep quite good said numerous surprising disappointment one bedroom suite balcony living room bedroom patio furniture chair table stool bathroom nicely fitted black granite pleasant amenity small hotel aspiration toilet shower tub combination single basin competitive market contrast batht ub ritz enjoy skyline view example think bottle water standard hotel caliber none breakfast food plentiful imaginative caper buffet served disappointing room loud scolded hostess arriving minute closing surprising season hotel plan arriv e hotel convenience location block orchard road walking distance botanic garden hyatt regency experience great hote l look forward quick remedy issue', 'great location stayed sw hotel night march hotel situated authentic busy chinato wn fine italian restaurant north beach hotel appeared deserted entire time think saw guest look like recently remodel ed good standard nice chinese style furniture decor room little small horrible view clean practical wife enjoyed short walk fisherman wharf surrounding area eatery choice quite excellent parking interesting way lot underneath hotel re al problem getting bizaare parking lot seen overall hotel pretty pay', 'best place bali coming bali regularly year visited hotel island question le meridien best offer troubled past including fight farmer moved land built financial problem as ian crisis relevant place feel little run compared newer hotel island dated look staff nice n speak english food hotel res taurant standard place like season ritz carlton chedi club said good thing le meridien right tanah lot especially gorgeo us morning shrouded mist large ground garden lovely peaceful bali away crowd kuta certainly price right bear mind d eciding stay minute le meridien seminyak great shopping big restaurant la lucciola living room ku ta hotel people wan t hit night life shopping fine family n want leave resort golf lover']

⋆Rating 4 Sample Reviews:
['decent hotel great location friendly staff hotel easy friendly staff clean room good breakfast sense safety', 'apex righ t pleased apex standard room nice size n view quiet n bother little comment dark needed light room n detract enjoya ble stay pleased booked website breakfast included quite good saving loved area tower bridge tower london st kather ines dock right end street butler wharf really worth look london couple year astounded price tube use bus lot cheaper luckily weather nice walked really nice weekend n hesitate stay', 'fine place stay amsterdam wife week really fine stay

cozy room nice antique furniture excellent quiet location ample restaurant walking distance minute stroll central stati on nice breakfast daily warm host recommended', 'perfect quarter hotel husband child stayed iberville night mid june town family wedding husband grew sibling know quarter extremely great time iberville staff extremely pleasant deal agree minor complaint review yes housekeeping extremely slow overall excellent stay n ritz amenity ritz hotel spa exe rcise facilitites day room person night ca n beat price spa fabulous workout facility big state art staff n know hotel sta ying treat yes buffet breakfast discontinued manager hotel subject fee person adult room applied family member tow n wedding waived requested discounted valet parking fee reduced spot hotel old pretty good shape n greatest view t h floor suite n really important kid blast resistance pool pretty location perfect like walk bad cabbing driving definitely stay trip considering iberville suggestion manager arriving contact liking experience willing make stay memorable de alt patience respect took care']

⋆Rating 5 Sample Reviews:
['luxury valley wing stayed valley wing bedroomed suite fabulous moment driver met airport whisked hotel mercedes presented wife beautiful orchid better room decorated cool superb view large pool garden breakfast amazing nice ch oice great service pool large fitted landscaped garden ate japanese contemporary restaurant good food service issue stay keycard stopped working continuously', 'perfect returned day trip italy began stay palazzo niccolini al duomo per fect introduction hotel hidden gem sight duomo perfect location getting city staff decor breakfast highly recommend treasure ca n wait', 'paradise value money paradise value money travelled bavarro stayed ifa villa resort accomodati on block basic clean good min walk beach lovely quiet food load restaurant buffet mexican international japenese bo ok day non buffet one booking start pm advice certain nationality started queuing pasta chef buffet great beach whit e sand palm tree went tropical storm cattamaran great entertainment good fun food plenty cuba libre definetely retu rn paid week inclusive n actually need spend money tip tropical storm trip', 'excellent hotel hotel great bargain staff fr iendly helpful speak english room quite small european standard impeccably clean maid nice clean room afternoon w ant pay extra euro upgrade superior deluxe room little larger hotel location excellent highly recommend']

TRAIN TEST SPLITTING:

The imbalanced dataset was divided into training and testing sets to prepare the data for model building and evaluation. The train_test_split() function from scikit-learn was used for this purpose.

Code:

```
from sklearn.model_selection import train_test_split

X = df_imbalanced['reviews']
y = df_imbalanced['overall']
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    stratify=y,
    random_state=42,
)
```

- The data was split in an 80:20 ratio, meaning 80% of the samples were used for training and 20% for testing.
- The parameter stratify=y ensures that the class proportions remain consistent in both the training and testing sets, preserving the imbalanced structure of the original dataset. The random_state=42 parameter guarantees reproducibility of the split.

Output:

```
Training samples: 7989
Testing samples: 1998

Class distribution in train set:
overall
5    2400
4    2000
3    1589
2    1200
1     800
Name: count, dtype: int64

Class distribution in test set:
overall
5    600
4    500
3    398
2    300
1    200
Name: count, dtype: int64
```

TF-IDF Vectorization:

Transform cleaned text reviews into numerical feature vectors that represent the importance of each word in a review relative to the entire dataset, making them suitable for machine learning models.

Code:

```
from sklearn.feature_extraction.text import TfidfVectorizer

# Vectorize text data
vectorizer = TfidfVectorizer(max_features=20000, stop_words='english')
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)

print("Vectorized X_train shape:", X_train_vec.shape)
print("Vectorized X_test shape:", X_test_vec.shape)
```

Output:

```
Vectorized X_train shape: (7989, 20000)
Vectorized X_test shape: (1998, 20000)
```

TF-IDF Equation

For a term t in document d in a corpus D:

$$\text{TF-IDF}(t,d,D)=\text{TF}(t,d)\times\text{IDF}(t,D)$$

Term Frequency (TF):

$$\mathrm{TF}(t,d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d}$$

Inverse Document Frequency (IDF):

$$\mathrm{IDF}(t,D) = \log \frac{N}{1 + |\{d \in D : t \in d\}|}$$

N → total number of documents in the corpus

|{d ∈ D : t ∈ d}| → number of documents containing term *t*

Interpretation:

- High TF-IDF → term is frequent in the document but rare in others (important).
- Low TF-IDF → term is common in all documents or rare overall (less important).

MACHINE LEARNING MODEL TRAINING AND EVALUATION

Two machine learning models Logistic Regression and Random Forest were selected to train on the imbalanced dataset. The models were initialized using scikit-learn with specific parameters: Logistic Regression was configured with a maximum of 1000 iterations to ensure convergence, while Random Forest was built with 300 estimators and the class_weight='balanced' parameter to handle class imbalance during training.

Code:

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score,classification_report


#initialize models
models = {
   "Logistic Regression": LogisticRegression(max_iter=1000, random_state=42),
   "Random Forest": RandomForestClassifier(n_estimators=300,
class_weight='balanced',random_state=42)
}

#training and evaluation
results = {}
```

```
for name, model in models.items():
    print(f"\n Training {name}...")
    model.fit(X_train_vec, y_train)
    preds = model.predict(X_test_vec)

    acc = accuracy_score(y_test, preds)
    print(f"✅ Accuracy for {name}: {acc:.4f}")
    print(classification_report(y_test, preds))

    results[name] = (model, acc)
```

Output:

```
Training Logistic Regression...
✅ Accuracy for Logistic Regression: 0.5591
              precision    recall  f1-score   support

           1       0.64      0.49      0.56       200
           2       0.46      0.42      0.44       300
           3       0.47      0.46      0.47       398
           4       0.52      0.51      0.52       500
           5       0.66      0.75      0.70       600

    accuracy                           0.56      1998
   macro avg       0.55      0.53      0.54      1998
weighted avg       0.56      0.56      0.56      1998


 Training Random Forest...
✅ Accuracy for Random Forest: 0.5035
              precision    recall  f1-score   support

           1       0.66      0.49      0.57       200
           2       0.57      0.27      0.37       300
           3       0.48      0.32      0.38       398
           4       0.43      0.36      0.39       500
           5       0.51      0.86      0.64       600

    accuracy                           0.50      1998
   macro avg       0.53      0.46      0.47      1998
weighted avg       0.51      0.50      0.48      1998
```

choosing best model:

```
best_model_name = max(results, key=lambda k: results[k][1])
best_model = results[best_model_name][0]

print(f"\n Best model: {best_model_name}")
```

Output:

```
Best model: Logistic Regression
```

Model and Test Data Saving:

After training and evaluation, the best-performing model from the imbalanced dataset (Model A) and its corresponding TF-IDF vectorizer were saved using the joblib library. The model was stored as model_A.pkl, and the vectorizer as vectorizer_A.pkl to enable easy reuse for predictions or further analysis without retraining.

Code:

```
import joblib
joblib.dump(best_model, "model_A.pkl")
joblib.dump(vectorizer, "vectorizer_A.pkl")

joblib.dump((X_test, y_test), "imbalanced_test.pkl")
```

CLICK HERE TO VIEW THE CROSSTEST AND STREAMLIT APP IMPLEMENTATION DOCUMENTATION